Gonito.net – open platform for research competition, cooperation and reproducibility

Filip Graliński*, Rafał Jaworski*, Łukasz Borchmann†, Piotr Wierzchoń†

Adam Mickiewicz University
*Faculty of Mathematics and Computer Science
†Institute of Linguistics
*ul. Umultowska 87, 61-614 Poznań, Poland
†al. Niepodległości 4, 61-874 Poznań, Poland
{filipg,rjawor,wierzch}@amu.edu.pl, borchmann@rainfox.org

Abstract

This paper presents the idea of applying an open source, web-based platform – Gonito.net – for hosting challenges for researchers in the field of natural language processing. Researchers are encouraged to compete in well-defined tasks by developing tools and running them on provided test data. The researcher who submits the best results becomes the winner of the challenge. Apart from the competition, Gonito.net also enables collaboration among researchers by means of source code sharing mechanisms. Gonito.net itself is fully open source, i.e. its source is available for download and compilation, as well as a running instance of the system, available at `gonito.net`. The key design feature of Gonito.net is using Git for managing solutions of problems submitted by competitors. This allows for research transparency and reproducibility.

1. Introduction

The field of Natural Language Processing struggles with numerous problems regarding research reproducibility and reuse. A common practice for the researchers is to focus on a very specific problem and engineer software that tackles it. Continuation of research is understood as further engineering of the same software for the same basic problem, with minor changes to the requirements. Even if such a project is carried out by a group of researchers rather than an individual, it should still be viewed as isolated. At the end of the project, the software might be released as an autonomous tool (open source or closed). However, even if an NLP tool gets released, it is often seen as a black box for other researchers, who use it as a submodule for isolated projects of their own.

As a result, many NLP researchers invent solutions they claim are specific for their project, while in reality these solutions could easily be generalised. Furthermore, when working on their projects, researchers tend to "reinvent the wheel" (even when working on the same data set). Instead of taking advantage of someone else's work they start from scratch and try to solve previously solved problems.

One effort to break the effect of isolation of NLP projects is organising shared tasks. In this scenario numerous researchers work on one specific, very well defined problem. Their work is organised as a competition, which is a highly motivating factor. However, the participants' work is still isolated, as they share experiences only after the shared task results are published.

This article focuses on a modified version of the shared tasks approach. It consists in organising shared tasks/challenges where the results and solutions submitted by participants could be open and ready to inspire other participants. The whole process is administered by the *Gonito.net* platform – custom made, open source software.

Similar platforms: Kaggle, CodaLab and others are described in Section 2. Section 3. lists related work examples. The Gonito.net platform itself is presented in detail in Section 4. Section 5. is a Gonito.net walkthrough. Lastly, ideas for improving citation of resources and tools and final conclusions are formulated in Sections 6. and 7.

2. Similar platforms

2.1. Kaggle

Kaggle is a commercial platform for organising competitions in the area of data science and corpus research, including natural language processing. It is a meeting point for business enterprises and data scientists, who are willing to tackle specific problems suggested by the enterprises.

In a typical Kaggle usage scenario, a large corporation proposes a problem, whose solution would be beneficial for its cause. In order to solve the problem, a significant amount of data has to be processed, using, potentially, a wide variety of methods of data analysis. The corporation provides the data and offers a tempting monetary prize (usually between 10 000 – 100 000 US dollars) for the scientist who is able to achieve the best analysis results. For instance, one of the active challenges in February 2016 consisted in predicting the relevance of search results on the *homedepot.com* website and offered a 40 000 US dollars prize. There are also challenges with no monetary prizes at all.

However, while submitting problem solutions is free of charge, proposing the problem (referred to as "hosting a challenge") is not. For that reason, subjects who host the challenges are referred to as "sponsors". Limitations in hosting challenges (with monetary prizes or without) are to ensure that only important, real-life problems are solved at Kaggle. Nevertheless, Kaggle also has a separate module – "Kaggle in class" – designed specifically for academic institutions. In this module, a university teacher can host a challenge for his or her students, without having to pay for that service (with some formal limitations).

As Kaggle deals with real problems and, more importantly, with real money (often considerable amounts), the platform implements a carefully designed system of scoring the users in the challenges and generating leaderboards. This is done mainly to ensure the fairness of the competition. Firstly, there is a distinction between *public* and *private* leaderboards. Users' submissions are first evaluated on development sets. These results are published in the public leaderboards. The submissions are evaluated on test sets, which are not made available for the users during their work. The results achieved on the test sets are used to generate the private leaderboards. These private leaderboards are made public only after the end of the challenge and serve to determine the final ranking. This is to prevent the users from overfitting their models to the test data.

To summarise, Kaggle incorporates many valuable concepts, such as the idea of competition between users or the system of scoring and generating leaderboards. However, the main drawback of Kaggle is the fact that it is a closed, commercial system and the cost of hosting challenges is often too high for individual researchers.

2.2. CodaLab

CodaLab (available at `codalab.org`) is a platform for hosting machine learning challenges, following similar principles as Kaggle. The main difference, though, lies in the fact that at CodaLab both entering and hosting a challenge are free of charge. Furthermore, the whole platform is open.

From a competitor's point of view, work on a challenge consists of:

1. downloading data for analysis from CodaLab in a so-called "competition bundle",
2. developing analysis tools,
3. wrapping results in a "reference bundle",
4. uploading the bundle to CodaLab.

Bundle is CodaLab's format for data interchange. Technically, it is a zip archive of a specific file structure. An example competition bundle is shown in Figure 1: `reference.zip` is the bundle storing the results, whereas `program.zip` is a bundle holding the sources of the program that generated the results.

Figure 1: CodaLab's competition bundle

```
competition.zip
  |- competition.yaml
  |- data.html
  |- evaluation.html
  |- logo.jpg
  |- overview.html
  |- program.zip
  |- reference.zip
  |- terms_and_conditions.html
```

While the openness of the platform is a significant advantage, the use of bundles for data interchange might not be comfortable. It would require a separate tool just to handle the bundle packing and unpacking process. Furthermore, CodaLab does not use any version control system to track the changes in submitted programs.

2.3. DrivenData

There is a competition hosting platform very similar to Kaggle, called DrivenData (`drivendata.org`). It is also a closed system, but it enables non-profit organisations to host challenges. Still, the platform is not well suited for academic purposes, as the challenges must bring solutions to practical problems of the hosting organisations.

2.4. Numerai

Numerai is a service available at `numer.ai`, hosting a worldwide tournament for machine learning special-

ists. The sole task in the tournament is predicting the stock market. The competition is possible thanks to the fact that Numerai provides anonymised stock market data and makes it publicly available. Top users are rewarded with money prizes.
It is fair to say that Numerai succeeded in organising a world-wide competition for data scientists. However, Numerai is not applicable to scientific research, as the platform is completely closed and cannot be used for any tasks other than stock market prediction.

3. Related work examples

The problem of research reproducibility has already been addressed in various ways. Some of the existing solutions to the problem use Git as the core system for managing development workflow. This section presents some of the most interesting setups, ensuring full reproducibility of the results, tidiness of resource storage and the transparency of the research.

3.1. Git-based setup

(Ram, 2013) describes a research workflow managed by the Git version control system. The idea of applying Git for this task is inspired by well-known software engineering findings, stating that version control systems (VCS) are crucial for managing resources especially in environments with multiple developers (see (Spinellis, 2005)). The primary functionality of a VC system is tracking changes in text files. Each change made by the developers in a file is bound with an informative comment, thus providing Git with exhaustive file history data. Not only does the history contain all text changes, but the reason for each change can be inferred from the comments.
Another distinctive feature of Git are branches. While all changes of the file are typically tracked in one history timeline (called the `master` branch), it is possible to create multiple branches. When a new branch is created, all changes to the file can be tracked either in the master branch, or the newly created branch. At any time, all changes from the new branch can be merged into the main branch. The branching mechanism is particularly useful when a developer needs to modify the file in a way that might make the file unusable for collaborators. In this case, the developer makes all the necessary modifications in his or her own branch and only after the changes are thoroughly tested, are they merged with the main branch.
Here are example use cases of the Git setup described in (Ram, 2013):

- creating lab notebooks, edited by multiple users,
- facilitating collaboration with the use of branches,
- backup and failsafe against data loss,
- freedom to explore new ideas and methods,
- increased transparency and verifiability.

3.2. Git and org-mode based setup

(Stanisic et al., 2015) describe more advanced ideas on preparing environment for reproducible research. By taking advantage of the Git branching feature, the authors developed their own branching model and defined typical operations on it. As Git is best suited for tracking changes in text files, the authors enhanced the environment with the use of the org-mode software, enabling task managing and organising work by commands written in plain-text format.
With the use of this environment, the authors managed to publish a fully reproducible paper on parallel computing: (Stanisic et al., 2014). However, although the Git repository itself is made publicly available, it is not ready for direct introduction to other research workflows.

4. The Gonito.net platform

The Gonito.net web application is an open-source platform for machine learning competitions. The idea is simple, yet powerful and malleable:

- challenges are published on the platform (at least as test sets, of course training and development sets could be provided as well),
- users can submit their solutions (the system output for the test set, possibly accompanied with the source codes or even full papers describing the system),
- the results are automatically evaluated by the system.

The results can then be tracked in terms of:

- timeline (who submitted what and when?),
- performance (which solution is the best at the moment? i.e. a leaderboard is presented),
- provenance (what is based on what? what was forked from what?).

The Gonito.net system is founded on two key (and interdependent) principles:

Be open
- Gonito.net is available[1] as open-source software under GNU Affero General Public License.

[1] `git://gonito.net/gonito` or `http://gonito.net/gitlist/gonito.git`

- Anyone can set up their own instance of Gonito.net (whether local or not) and run whatever challenges they please.
- Users are encouraged (but not forced) to share the source codes of their solutions.
- Users are free to use whatever programming language and tools to generate the output (only Git, a widely adopted tool, is required).

Use Git

- Solutions can be uploaded to the Gonito.net platform with Git without clicking around and uploading files in a browser.
- New challenges are created as Git repositories.
- With Git it is possible to track which solution was forked and reused in another solution.
- Even if a Gonito.net platform ceases to exist, the results and source codes may still be available as a regular Git repository to be cloned and inspected with standard Git tools, no external database is needed.

The first principle differentiates Gonito.net from Kaggle, whereas the second one – from CodaLab.
Notwithstanding the simplicity of the idea, Gonito.net can support a variety of workflows and could be used for a wide range of different purposes:

- as an auxiliary teaching tool (for machine learning or NLP classes) helping to keep track of students' assignments and progress (just as in "Kaggle in class", but students' identity and work do not have to be shared with any external company – if privacy is a concern),
- within a company when working on a machine learning problem,
- within a small group of researchers to keep track of the progress (e.g. when working on a paper),
- for organising shared tasks,
- for tracking effort of a given research community on standard tasks in a longer-term perspective (not just as a one-off event).

Actually, a Gonito.net challenge can go through a combination or sequence of such stages: it can be started and "test-run" as a student course assignment, then a small group of researchers could work on it locally and after some time the challenge could be made public as a shared task competition, but even when the shared task is completed, the Gonito.net platform could be used to continuously track the progress of the whole research community on a given problem. If adopted, Gonito.net could provide the focal point where the best solution for a particular problem might become common knowledge ("everyone knows that everyone knows... that the current best result is there").
A Gonito.net challenge does not have to be only about the competition – an appropriate blend of competition and cooperation could be achieved with Gonito.net, as Git makes it easy to reuse other peoples' solutions and build on them. (Just as science, in general, is a mixture of "racing to the top" and "standing on the shoulders of giants".)
Gonito.net is written in Haskell (using the Yesod web framework) and is accompanied with *GEval*, a library and stand-alone tool for machine learning evaluation. At this time, the accuracy, (root-)mean-square error and BLEU metrics are implemented in GEval. Due to the application author's background in natural language processing the stress has been on NLP metrics and challenges so far. Nevertheless, Gonito.net could be used for any machine learning challenge, provided that an evaluation metric is implemented in GEval.
An instance is available at `http://gonito.net` along with some sample (but non-toy) NLP challenges (of course, anybody can set up another instance, whether it is to be publicly or only internally available). For example, one of challenges is about guessing the publication year (1814-2013) of a short Polish text.[2] Another challenge will be described in Section 5..
The sample instance is accompanied with a Git-hosting system (Gitolite + GitList). The whole assembly is packaged and made available as a virtual machine image so that anybody could get it up and running quickly if it is to be self-hosted.

5. Gonito.net walkthrough

This section presents a walkthrough of participating in a challenge from a competitor's point of view. The challenge used in this walkthrough will be one of Gonito.net currently running challenges: the "He Said She Said Classification Challenge". The task is to identify, whether a Polish text is written by a male or female.
The corpus used in this task was obtained by processing the Common Crawl-based Web corpus of Polish (Buck et al., 2014), using the method described in detail in (Graliński et al., 2016). Author's gender

[2] http://gonito.net/challenge/retroc

determination was possible thanks to the existence in Polish of gender-specific first-person expressions (for instance, *I said* will be rendered in Polish as *powiedziałem* or *powiedziałam* depending on whether it was spoken or written by a man or a woman). Text fragments without such expressions were discarded. Then, a gender-neutral version of the corpus was prepared and made available on Gonito.net for the purposes of training and testing classifiers.

5.1. Prerequisites

Upon starting a challenge it is assumed that the competitor is familiar with Git and a Unix-like operating system environment.

The first step[3] of setting up Gonito.net's specific environment is installing the GEval software. It is written in the Haskell programming language and requires the Haskell Stack program, available for download at `https://github.com/commercialhaskell/stack`. With Haskell Stack installed, the following commands install GEval:

```
git clone git://gonito.net/geval
cd geval
stack setup
stack install
```

The simplest way to get the challenge data is to clone the read-only repository, e.g. for the "He Said She Said" challenge the URL is `git://gonito.net/petite-difference-challenge` (also reachable and browsable at `http://gonito.net/gitlist/petite-difference-challenge.git/`). Then a user can use his or her own repository, wherever it is hosted (at his or her private server, at GitHub, etc.). In sections 5.2.–5.8., however, a slightly more complicated (but recommended in the long term) workflow using repositories hosted at Gonito.net is presented.

5.2. Getting the work repository

Every user registered at Gonito.net receives their own Git repository for each challenge they might participate in. In order to start working, the user must first enter his or her login name and a SSH public key at `http://gonito.net/account` and then clone the repository. Assuming the user login name is `tom` and challenge ID: `petite-difference-challenge` (actual ID of the "He Said She Said" challenge), the following commands are used to prepare the user's repository:

```
git clone ssh://gitolite@gonito.net/
  tom/petite-difference-challenge
cd petite-difference-challenge
git pull ssh://gitolite@gonito.net/
  petite-difference-challenge
git push origin master
```

Note that the first command clones an empty repository – the private repository of the user `tom`. The third command is used to pull the contents of the mother repository, containing the challenge data and solution template.

The solution template contains the following files and directories:

- `README.md` – text file with a detailed challenge description,
- `train` – folder with training data, usually compressed plain text file in tab separated values (TSV) format,
- `dev-0` – folder with annotated corpus, containing input and expected data (files `in.tsv` and `expected.tsv` respectively),
- `test-A` – folder with test corpus, containing only input data.

(More development and test sets may be added later by the challenge organisers.)

5.3. Working on solution

While working on a solution, a user is required to develop a data analysis tool able to produce predictions for data in the `dev-0/in.tsv` and `test-A/in.tsv` files. These predictions should be stored in the `dev-0/out.tsv` and `test-A/out.tsv` files respectively. With the help of the GEval software, a user is able to evaluate his or her results on the development set locally, before submitting them to Gonito.net. In order to do this, the following command[4] must be issued from within the top directory of the solution repository:

```
geval --test-name dev-0
```

[3] Actually, this step is not obligatory, as a user could just use the web application to evaluate his or her submissions. On the other hand, it is recommended to check the solution locally on the development set before submitting to the web application in order to make sure it is correctly prepared and formatted.

[4] It is assumed that `~/.local/bin` is added to the `$PATH` environment variable.

In the "He Said She Said" challenge, the result is one number representing the accuracy of guesses.
When a user is satisfied with the performance of the software on the development set, he or she may try to submit the solution to Gonito.net and check the results on the test set.

5.4. Submitting a solution

Submitting a solution to Gonito.net consists of two steps – sending output files via Git and notifying Gonito.net of the changes in the repository.
In order to send the output files via Git one should issue the following commands:

```
git add dev-0/out.tsv test-A/out.tsv
git commit -m 'my brilliant solution'
git push origin master
```

In the second step, the user must click the "Submit" button[5] in Gonito.net and provide the following information:

- informative submission description,
- repository URL (auto-filled to the URL of the user's repository on Gonito.net),
- repository branch (auto-filled to `master`).

Alternatively, instead of manually clicking the "Submit" button, a Git "hook" can be configured to get Git to notify Gonito.net of a commit and to trigger an evaluation automatically (just as continuous integration servers are often configured to be triggered whenever a new commit is pushed).
All submissions for a challenge are evaluated on both the development and test set. All these results are visible to all participating users (the user names could be anonymised if desired).

5.5. Best practices

Although it is not formally required, the users are encouraged to submit the sources of their programs along with the results via their Git repositories. A good idea is to provide a Makefile which could facilitate the process of running the program on the data. However, the recommendation for submitting the sources of the program excludes statistical models and any other resource files that can be generated by the software. (The authors of Gonito.net are planning incorporating git-annex Git extension into the Gonito.net setup, so that models and other large files could be uploaded and downloaded there, cf. (Korolev and Joshi, 2014)). Furthermore, if the tool uses randomisation at any step of the analysis, this randomisation should use a specified seed number.
All these recommendations help to ensure transparency and reproducibility of the solutions.

5.6. Example solutions to the "He Said She Said" challenge

Let us now prepare a simple, baseline solution for the "He Said She Said" problem. The solution will always guess that the text is written by a male. Sample code for this solution can be given as a shell script:

```
#!/bin/sh

filename=$1/in.tsv

while read -r line
do
    echo M >> $1/out.tsv
done < "$filename"
```

Let us also prepare a Makefile:

```
all: classify-dev classify-test geval

classify-dev: dev-0/out.tsv

classify-test: test-A/out.tsv

dev-0/out.tsv:
  ./classify.sh dev-0

test-A/out.tsv:
  ./classify.sh test-A

clean:
  rm -f dev-0/out.tsv test-A/out.tsv
```

Now, after issuing the commands `make clean` and `make` we obtain the files `dev-0/out.tsv` and `test-A/out.tsv`. Checking the results achieved on development set (`geval --test-name dev-0`) gives the expected result of 0.5 (as the test corpora are evenly balanced in terms of the number of male and female texts).

5.7. Availability and reproducibility of a submission

This submission described in Section 5.6. was made public as commit 86dd91 and, consequently:

- in the electronic edition of this paper, the above commit number prefix

[5] See `http://gonito.net/challenge-submission/petite-difference-challenge`

is clickable[6] as `http://gonito.net/q/86dd914ad99a4dd77ba1998bb9b6f77a6b076352` and leads to a submission summary,

- the submission (both the output data and source codes) is available as `git://gonito.net/petite-difference-challenge` (branch `submission-00072`) – anybody can clone the repository, inspect the data and source codes, try to reproduce the results (with `make clean && make`) and create their own solution based on them,
- the submission is accessible with a Web browser at `http://gonito.net/gitlist/petite-difference-challenge.git/submission-00072/`,
- as Git commit ID (SHA-1 hash) is used to identify a submission, the submission data might be available even if Gonito.net stops working (provided that it was pushed to some external repository, e.g. GitHub).

5.8. Non-trivial submission

A less trivial approach can be implemented in Python with scikit-learn machine learning library (Pedregosa et al., 2011), and be designed as follows:

1. convert text to a matrix of token counts, previously lower-casing it, and stripping all the punctuation,
2. make term-document matrix from joined punctuation marks' n-grams,
3. concatenate results of the above transformer objects with `FeatureUnion`,
4. transform the above count matrix to a normalised TF-IDF representation,
5. train logistic regression classifier.

Such a solution received an accuracy score above 0.63 and is available with the source code on the Gonito.net (commit 2cb823{model.py}).
The leaderboard with the graph presenting the submission times (x-axis) and the scores (y-axis) is given in Figure 2. Each point represents one submission and the relations between submissions (what was forked from what?) are represented with arrows.

[6]Alternatively, the ID `86dd91` could be entered manually at `http://gonito.net/q` or directly pasted into a link (`http://gonito.net/q/86dd91`)

6. Improving citation of tools and resources

A very simple, technical measure within the Gonito.net platform to improve citation of tools and resources is to introduce a standard location for a file (`references.bib`) where BibTeX references to the paper describing the solution (and the solutions forked and re-used so far) are kept. The file should be initiated with the reference to the paper describing the challenge itself and the related language resource. With each solution to be described in some paper, the `references.bib` file should be extended with a new entry.
This approach, though, has a weakness: the BibTeX entry is, obviously, available only when the paper is published, which is usually much later than the submission to Gonito.net is done. Actually, it is a more general problem: the final paper itself will usually be completed *after* the submission (even if it is written along with the software solution and some draft version is ready then). The workflow we propose is to push a new commit with the final paper and another one with the BibTeX entry, when they are ready and submit them to Gonito.net (with *the same* output data as the initial submission). The idea is that Gonito.net stores an extra check-sum of output files (just output files, excluding source codes) and the commits at various stages (but related to the same results) can be tracked by Gonito.net and presented together.

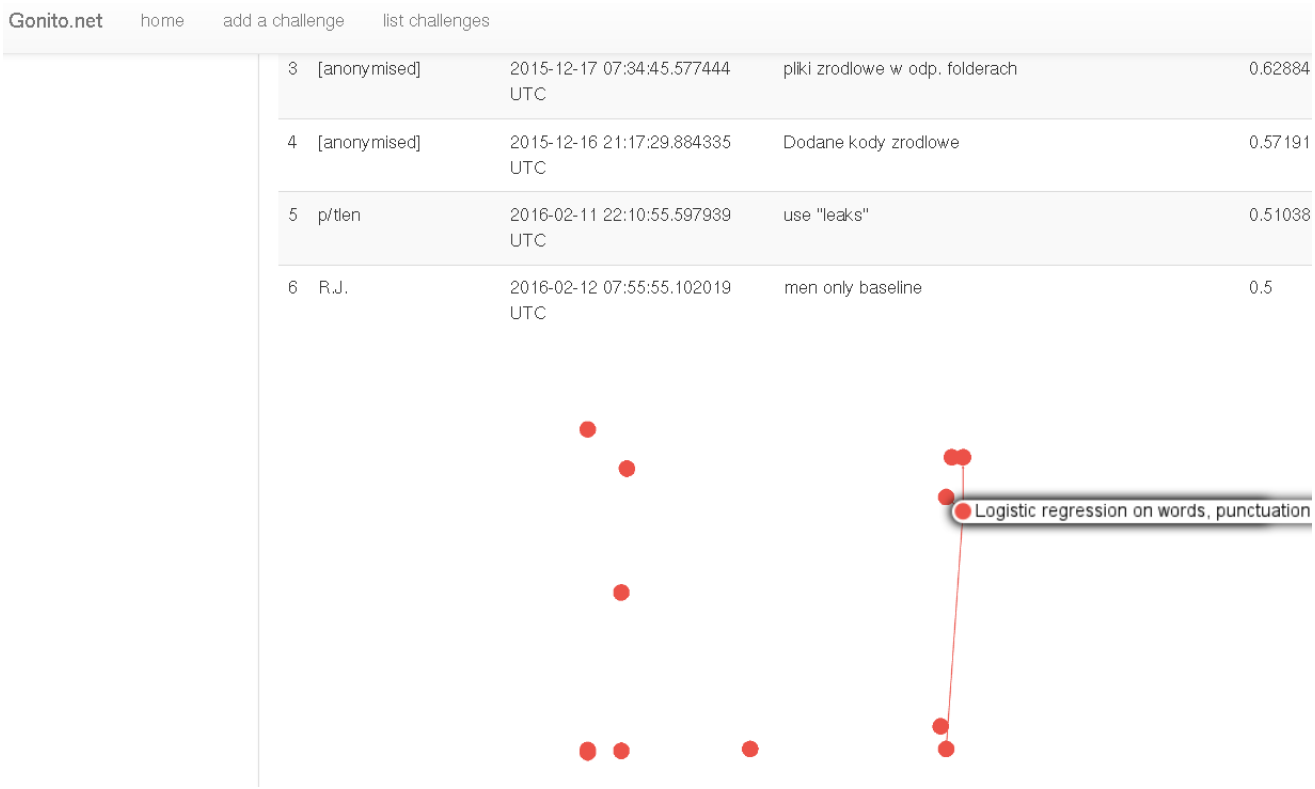
Acknowledgements

Work supported by the **Polish Ministry of Science and Higher Education** under the **National Programme for Development of the Humanities**, grant 0286/NPRH4/H1a/83/2015: "50 000 słów. Indeks tematyczno-chronologizacyjny 1918-1939".

7. Conclusions and future work

Gonito.net is currently hosting three challenges, being tackled by a small community of researchers and students at an academic institution. With this article we suggest that other researchers try to apply Gonito.net for hosting NLP or other types of scientific tasks. Importantly, the openness of Gonito.net makes it applicable in any research workflow. After the first months of using the platform it is possible to conclude that Git proves extremely useful in managing the process of storing and exchanging of tools and resources.
Ideas for future work include further increasing the transparency of submitted solutions by providing mechanisms facilitating code reuse. Ideally, a new solution for a problem could be based on the current best solution. The researcher should try to improve the best

Figure 2: Gonito leaderboard with a graph representing the submissions and the relations between them



solution and bring the score to a higher level. Thus, the competition could become a productive cooperation.

8. References

Buck, Christian, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavk, Icelandik, Iceland, May.

Graliński, Filip, Łukasz Borchmann, and Piotr Wierzchoń. 2016. He said she said – male/female corpus of polish. In press, to appear in Proceedings of the LREC 2016 Conference.

Korolev, Vlad and Anupam Joshi. 2014. Prob: A tool for tracking provenance and reproducibility of big data experiments. *Reproduce'14. HPCA 2014*, 11:264–286.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ram, Karthik. 2013. Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, 8(1):1–8.

Spinellis, Diomidis. 2005. Version control systems. *Software, IEEE*, 22(5):108–109.

Stanisic, Luka, Samuel Thibault, Arnaud Legrand, Brice Videau, and Jean-François Méhaut. 2014. Modeling and Simulation of a Dynamic Task-Based Runtime System for Heterogeneous Multi-Core Architectures. In *Euro-par - 20th International Conference on Parallel Processing*, Euro-Par 2014, LNCS 8632, Porto, Portugal, August. Springer International Publishing Switzerland.

Stanisic, Luka, Arnaud Legrand, and Vincent Danjean. 2015. An effective git and org-mode based workflow for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):61–70, January.